CLAIMS

1. (Currently Amended) An apparatus comprising:

one or more processors; and

memory communicatively coupled to the one or more processors, the memory having stored thereon a plurality of instructions that, when executed on the apparatus, configure the one or more processors to implement:

a virtual machine ~~means~~, instantiated in managed code to execute with a runtime loader, for executing a first assembly and a second assembly ~~assemblies~~ of one or more files instantiated in the managed code;

the first assembly configured to make ~~means for making~~ a call for access ~~by the first assembly of one or more of the files instantiated~~ in the managed code to the second assembly ~~of one or more of the files instantiated~~ in the managed code at Just-In-Time (JIT) compilation time;

a JIT compiler for compiling each of the first assembly and the second assembly ~~assemblies~~ into native code for execution as native code, wherein during compilation, based upon a determination by a determining component that it is unknown whether the call from the first assembly to the second assembly should be permitted, the JIT compiler is configured to insert a runtime stub into the call before compiling the first assembly and the second assembly ~~assemblies~~;

an interceptor ~~means~~ for intercepting the call from the first assembly to the second assembly at a runtime; and

the determining component ~~means~~, based upon a user identification (ID) for at least one of the first assembly and the second assembly ~~assemblies~~ of the one or more files, for determining, at the runtime, access privileges of the first assembly of the one or more files to the second assembly of the one or more files.

2. (Currently Amended) The apparatus as defined in Claim 1, wherein the plurality of instructions configure the one or more processors to further implement ~~further comprising~~:

an execution engine, instantiated in a native code, to execute the virtual machine in runtime; and

an operating system in native code to be executed with one or more of the compiled first assembly and the second assembly ~~assemblies~~.

3. (Cancelled).

4. (Currently Amended) The apparatus as defined in Claim 1, wherein the determining component is configured to prevent ~~means for determining access privileges comprises: means for preventing~~ the access of the first assembly to the second assembly when the determination based upon the ID for at least one of the first assembly and the

second assembly ~~assemblies~~ is unfavorable based upon predetermined criteria for the respective IDs.

5. (Currently Amended) The apparatus as defined in Claim 1, wherein the determining component is configured to prevent ~~means for determining access privileges comprises; means for preventing~~ the access of the first assembly to the second assembly when the ID for the first assembly does not match the ID for the second assembly based upon a predetermined match criteria for the respective IDs.

6. (Currently Amended) The apparatus as defined in Claim 1, wherein the determining component is configured to prevent ~~means for determining access privileges comprises: means for preventing~~ the access of the first assembly to the second assembly when the first assembly is in a first application domain and the second assembly is in a second application domain, and the first and second application domains do not match based upon a predetermined match criteria for application domains.

7. (Currently Amended) The apparatus as defined in Claim [[3]] 2, wherein:

the determining component is further configured to permit ~~means for determining access privileges comprises means for permitting~~ the access of the first assembly to the second assembly when the ID for the first assembly matches the ID for the second assembly based upon a predetermined match criteria for the respective IDs; and

the plurality of instructions configure the one or more processors of the apparatus to further implement ~~the apparatus further comprises~~:

~~means for loading the native code with~~ a Common Language Runtime (CLR) loader in the native code portion to load the compiled native code~~; and~~

~~means for executing the compiled native code in the native code portion, wherein the first assembly accesses the second assembly~~.

8. (Currently Amended) The apparatus as defined in Claim 1, wherein the determining component is configured to permit ~~means for determining access privileges comprises: means for permitting~~ the access of the first assembly to the second assembly when previous access to said second assembly by said first assembly had been permitted.

9. (Currently Amended) The apparatus as defined in Claim 8, wherein the previous access had been permitted following a prior determination that was favorable based upon a predetermined comparison criteria for the respective IDs.

10. (Currently Amended) The apparatus as defined in Claim 1, wherein the plurality of instructions configure the one or more processors to further implement ~~further comprising~~:

a verifying component ~~means~~, prior to determining access privileges, for verifying ~~determining~~ whether the ID is accurate for the first assembly and the second assembly ~~assemblies~~; and

wherein the determining component is configured to:

~~means, upon the determination by the accuracy means that either of said IDs is inaccurate, for:~~

permit the JIT compiler ~~permitting the means for compiling~~ to compile at least one of the first assembly and the second assembly ~~assemblies~~ into native code; and

delay determination of ~~delaying the means for determining~~ access privileges until the ID is accurate for the first assembly and the second assembly ~~assemblies~~.

11. (Currently Amended) The apparatus as defined in Claim 10, wherein the verifying component is further configured to verify ~~means is for further determining~~ that the ID is accurate for the first assembly and the second assembly ~~assemblies~~ at [[a]] the runtime for the native code.

12. (Currently Amended) The apparatus as defined in Claim 10, wherein the determining component ~~means for delaying the means for determining access privileges~~ is for further halting the delay at [[a]] the runtime for the native code.

13. (Original) The apparatus as defined in Claim 1, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language

compiler, are represented by the first and second assemblies in respective application domains.

14. (Currently Amended) The apparatus as defined in Claim [[3]] 2, wherein the execution engine ~~means~~ in the native code portion comprises a compiler to compile each said assembly into native code for execution by the native code portion.

15. (Currently Amended) The apparatus as defined in Claim [[3]] 2, wherein the execution engine ~~means~~ in the native code portion comprises:

a CLR loader to load the compiled native code for execution by the native code portion.

16-46. (Cancelled).

47. (Previously Presented) A server comprising:

a virtual machine, instantiated in managed code to execute with a runtime loader, to execute first and second assemblies of one or more files instantiated in the managed code, each of the first assembly and the second assembly being registered as a server object with the server;

a first module to make a call for access by the first assembly to the second assembly at Just-In-Time (JIT) compilation time;

an intercept module to intercept the call from the first assembly to the second assembly;

a second module, based upon user identification (ID) for at least one of the first and second assemblies, to determine access privileges of the first assembly to the second assembly; and

a JIT compiler module, based upon a first determination made at the second module that it is unknown whether the call from the first assembly to the second assembly should be permitted, to perform actions comprising:

inserting a runtime stub into the call; and

compiling the first assembly and the second assembly in the managed code into native code for execution as native code, wherein at runtime when the native code of the first assembly and the second assembly is executed at the server, the second module of the server is configured to make, based upon the user ID for each of the first assembly and the second assembly at the runtime, a second determination of whether the call by the first assembly to the second assembly shall be permitted at the runtime.

48. (Currently Amended) The server ~~apparatus~~ as defined in Claim 47, further comprising:

an execution engine, instantiated in a native code, to execute the virtual machine in runtime; and

an operating system in native code to be executed with one or more of the compiled first and second assemblies.

49. (Currently Amended) A method implemented by a computing device, the method comprising the steps of:

identifying a cross assembly call from a first assembly of one or more of the files instantiated in the managed code to a second assembly of one or more of the files instantiated in the managed code at a Just-In-Time (JIT) compilation time, wherein access privilege of the cross assembly call has not been verified based upon an identification (ID) for at least one of the first assembly and the second assembly ~~assemblies~~ of the one or more files;

making, via the computing device, a first determination at the JIT compilation time that it is unknown whether the call from the first assembly to the second assembly should be permitted, wherein the first determination is based upon the ID for at least one of the first assembly and the second assembly ~~assemblies~~;

inserting a runtime stub to the cross assembly call in the managed code to postpone a verification of the cross assembly call at the JIT compilation time, the runtime stub being configured to be used to make a call back at a runtime;

compiling the first assembly and the second assembly ~~assemblies~~ in the managed code into native code for execution as native code;

intercepting the cross assembly call at the runtime; and

making, via the computing device, a second determination at the runtime to decide, based upon the ID for at least one of the first and second assemblies at the runtime, whether the call by the first assembly to the second assembly shall be permitted.

50. (Currently Amended) [[A]] The method as recited in claim 49, further comprising preventing the access of the first assembly to the second assembly when the second determination based upon the ID for at least one of the first assembly and the second assembly ~~assemblies~~ is unfavorable based upon predetermined criteria for the respective IDs.

51. (Currently Amended) **[**[A]] The method as recited in claim 49, further comprising preventing the access of the first assembly to the second assembly when the ID for the first assembly does not match the ID for the second assembly based upon a predetermined match criteria for the respective IDs.

52. (Currently Amended) **[**[A]] The method as recited in claim 49, further comprising preventing the access of the first assembly to the second assembly when the first assembly is in a first application domain and the second assembly is in a second application domain, and the first and second application domains do not match based upon a predetermined match criteria for application domains

53. (Currently Amended) **[[**A]] The method as recited in claim 49, further comprising permitting the access of the first assembly to the second assembly when the ID for the first assembly matches the ID for the second assembly based upon a predetermined match criteria for the respective IDs.

54. (Currently Amended) **[[**A]] The method as recited in claim 49, further comprising permitting the access of the first assembly to the second assembly when previous access to said second assembly by said first assembly had been permitted.

55. (Currently Amended) **[[**A]] The method as recited in claim 49, further comprising launching a Common Language Runtime (CLR) loader to the first assembly and the second assembly ~~assemblies~~ in managed code.

56. (Currently Amended) [[A]] The method as recited in claim 49, further comprising verifying that the ID is accurate for the first assembly and the second assembly ~~assemblies~~.

57. (Currently Amended) [[A]] The method as recited in claim 56, wherein the step of verifying comprises verifying that the ID is accurate for the first assembly and the second assembly ~~assemblies~~ at the runtime.

58. (Currently Amended) One or more computer readable storage media ~~medium~~ having stored thereon a plurality of instructions that, when executed [[on]] by a computing device having one or more processors, cause the one or more processors to implement the method as recited in claim 49.